Introduction

This technical white paper describes how to use the MySQL `flush tables with read lock` feature with Compellent Data Instant Replay in a Linux environment for clones and backups. The `flush tables with read lock` feature is used to leverage SAN vendors snapshot technologies to create a mirror of production volumes.

This paper focuses specifically on the following topics:

- Using the `flush tables with read lock` feature to leverage Compellent Replay functionality to perform the following:
  - Database Clones
  - Database Backups

Audience

The primary target audience for this technical white paper is experienced MySQL Enterprise database administrators, Linux system administrators, SAN storage administrators, and architects who analyze, design, and maintain robust database and storage systems. Readers should be familiar with MySQL Enterprise, Linux Operating System, and Compellent products.

CML Data Instant Replay Overview

Compellent provides unlimited space-efficient snapshots with instantaneous recovery to any server. Traditional solutions only provide a limited number of snapshots. Traditional snapshots consume a significant amount of space and can have a negative impact on performance. This is because typical snapshot implementations utilize a copy-on-write technology. As the volume changes, they need to make

Disclaimer:
Information in this document is considered informational only. It is deemed unstable and dynamic, and is intended for support purposes only. It is subject to change without notice.

© 2004 Compellent Technologies. All rights reserved. Reproduction in any manner without the express written permission of Compellent Technologies is strictly prohibited. Trademarks used in this text are property of Compellent Technologies, or their respective owners.
copies of that volume to reflect the incremental changes made to that volume. That consumes a lot of disk space and can be difficult to manage. In most complete systems, to recover a snapshot, an administrator must make a full volume clone of the volume.

Compellent’s solution is Data Instant Replay. Data Instant Replay offers an unlimited number of space-efficient snapshots, or Replays, based on any time interval. Automatic coalescence allows you to consolidate expired replays into less frequent intervals and recovers disk space. Because administrators don’t need to create a copy or clone of the original volume, recovering a Replay is extremely fast. Replays can be mapped to any server at any time in just a few seconds.

MySQL Flush Tables with Read Lock Feature

The `flush tables with read lock` command closes all open tables and lock all tables for all databases with a read lock until you explicitly release the lock by executing `unlock tables` or you log out of your current session. This is a very convenient way to get backups if you have storage snapshot capability.

The `flush tables with read lock` command work with any MySQL engine, because this solution is fully based on volume level. Some engines have their own online backup solution – like the InnoDB engine has InnoDB Hot Backup. This technical white paper, however, is written for MySQL engines that do not have an online backup solution.

Remember that CML replays operate at the volume level. When taking a replay of a volume, any data in that volume will be in that replay. If you have multiple databases in one volume, then multiple databases will be part of that replay.

Cloning MySQL Database

The following example shows how you can utilize CML Data Instant Replay to clone a MySQL database server.

Assumption: Server Galaxy is the production server running Red Hat Linux with the below configuration. Server Falcon is the cloned host and has only boot volume.

- One volume was created for boot from SAN
- One volume was created for MySQL installation and data mounted under `/mysql`
- One volume was created for MySQL logs mounted under `/mysql/log`

```
<table>
<thead>
<tr>
<th>Filesystem</th>
<th>1K-blocks</th>
<th>Used</th>
<th>Available</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/sdc5</td>
<td>11935660</td>
<td>3543352</td>
<td>7776220</td>
<td>32%</td>
<td>/</td>
</tr>
<tr>
<td>/dev/sdc1</td>
<td>124427</td>
<td>18442</td>
<td>99561</td>
<td>16%</td>
<td>/boot</td>
</tr>
<tr>
<td>tmpfs</td>
<td>4088624</td>
<td>0</td>
<td>4088624</td>
<td>0%</td>
<td>/dev/shm</td>
</tr>
<tr>
<td>/dev/sdb5</td>
<td>51606108</td>
<td>773128</td>
<td>48211544</td>
<td>2%</td>
<td>/mysql</td>
</tr>
<tr>
<td>/dev/sda5</td>
<td>41284896</td>
<td>554848</td>
<td>38632900</td>
<td>2%</td>
<td>/mysql/log</td>
</tr>
</tbody>
</table>
```

**my.cnf file contains:**

```ini
basedir = /mysql/mysql
datadir = /mysql/data
log_bin = /mysql/log/binary-logs/galaxy-bin
relay_log = /mysql/log/binary-logs/galaxy-relay-bin
innodb_data_home_dir = /mysql/data
innodb_data_file_path = ibdata1:20M:autoextend
```
Steps for Creating Clones of the whole Server

Pre-requisite:

Do the following on your cloned host:

- Create mysql user and group
- Copy /etc/init.d/mysql.server from server Galaxy to server Falcon
- Copy /etc/my.cnf from server Galaxy to server Falcon

Run the below steps on production server (Galaxy) to create clone.

1. Connect to MySQL server and run `flush tables with read lock` command

```
mysql> flush tables with read lock;
Query OK, 0 rows affected (0.00 sec)
```

2. Create CML Replay. You can use the GUI to take replays or use the CLI. Do not exit the mysql session on step #1 until you finish taking replays because once you exit the mysql session, all locked tables will be released.

3. Unlock all tables. With the same mysql session from step #1, enter the `unlock tables` command.

```
mysql> unlock tables;
Query OK, 0 rows affected (0.00 sec)
```

4. Create CML views from the replays and map the views to server Falcon.

5. Once the views are mapped on server Falcon, mount the views under the same mount point as server Galaxy (/mysql and /mysql/log).

6. Start MySQL on server Falcon

```
/etc/init.d/mysql.server start --log-bin
```

If you want to script the create replay process, then you need to put the create replay command in a script and call this script inside the mysql session.

Here is an example. I have created a shell script called `cr_mysql_replay.sh`. This script contains the CML CompCU commands to create replays based on a specific configuration. Now I create a master script called `mysql_replay_master.sh` that contains the following code:

```
${SQLDIR}/mysql -u root <<EOF
flush tables;
system ${DIR}/replay/cr_mysql_replay.sh
exit
EOF
```
With this script, all tables will be unlocked automatically after the session is exited via the exit syntax. You can call this script at any time or create an entry in crontab.

Backing up MySQL Database

Backing up MySQL database is similar to cloning a database. To backup your database, follow step #1 to step #3 from the clone section above to create replays for backup. If you use replays for database recovery, you need to understand the recovery process of MySQL. Configure MySQL to use binary logs in order to do a roll forward recovery of your database. With binary logs, you can use an older data file from the previous replay and apply all the binary logs to roll forward. MySQL roll forward recovery process will not be discussed here.

Conclusion

With Compellent Data Instant Replay technology, you can create a clone or backup of your production MySQL Enterprise in a matter of seconds regardless of database size.